**checkin**:输入\*ctf{this\_1s\_flA9\_f0rm4t}

**re**:反汇编后，发现main读入一个字符串，并与0x601058处字符串翻转比较。查得0x601058处字符串为“}yawyna\_lahc\_esrever\_a\_si\_siht\_wonk\_uoy{ftc\*” ，翻转后为\*ctf{you\_know\_this\_is\_a\_reverse\_chal\_anyway}

**crypto**:扔到python跑一遍即可得到\*ctf{itworks!}

**blog: flag1**:博客主人名为shIr0kawa，到github搜索这个人，在个人简介中找到\*ctf{W3\_@rE\_GitHubbers!}

**login**:得到flag的要求是isLogin为true以及user为admin。观察代码可以发现，defaultValues的赋值方式是若不存在则声明并赋值。因此，手动在链接中加入isLogin=true就可以阻止初始化。同时，在cookie中加入一条passwd，值为任意值，避免isLogin被赋值为false。得到\*ctf{eXtrAct\_15\_NoT\_S@fe}

**pyc**:反编译check.pyc，内容为：

|  |
| --- |
| **def** do\_check**(**s**):**  **if** s**[:**5**]** **!=** '\*ctf{'**:**  **return** **False**  **if** s**[-**1**]** **!=** '}'**:**  **return** **False**  res **=** **[**  42**,** 46**,** 61**,** 60**,** 32**,** 62**,** 60**,** 40**,** 16**,** 46**,** 121**,** 107**,** 83**,** 112**,** 114**,** 118**,** 33**,** 43**,** 60**,** 50**,** 57**,** 59**,** 56**,** 0**,** 14**,** 61**,** 39**]**  **if** len**(**s**)** **>** len**(**res**):**  **return** **False**  accu **=** 0  **for** a**,** b **in** zip**(**s**,** res**):**  accu **^=** ord**(**a**)**  **if** accu **!=** b**:**  **return** **False**  accu **^=** 103  **return** **True** |

简单的顺序异或，逆推一下即可得到\*ctf{yes\_Y0u\_Dec0mpiled\_iT}

**diop**:将f对a取模，由于a及其巨大，x^2远远小于a，即有f % a == x^2。然后再求出y即可。得到\*ctf{Did\_you\_try\_sympy\_or\_solve\_by\_hand}

**www**:反编译后发现main函数读取信息后将给定位置的四个字节改为指定的内容。查看汇编指令时，发现main函数上方，0x804851D处有一个未被调用的后门函数，其内容为调用参数为”/bin/sh”的system函数。因此，只需要想办法让进程执行到这段函数即可。由于栈地址随机，难以通过更改ret地址的方式。观察到修改完后调用系统调用puts输出”bye”，调用方式为调用函数\_puts，它会jmp到0x804A00C所存储的地址处。而0x804A00C的内存是可写的，所以只要将这个地址改成后门函数的入口地址即可。有bash后访问/home/pwd1/flag得到\*ctf{writable\_got\_is\_helpful}

**dontcheat**:选出所有格子中颜色和其他不一样的格子。由于颜色差距很大，将显示器亮度和对比度调到最高然后手玩32层即可。得到CTF{Can\_Y0u\_5ee\_m2}

**CompCipher**:反编译后，发现得到key后，flag会被子程序0x4008AD加密然后输出。观察子程序的代码，其中有各种混淆。整理后发现实际上子程序的加密方法如下：首先将读入的整数和256取模置于加密用数组0x6020E0[0]位置（之后简称为数组X），接下来令X[i]的值为X[0] + i。然后利用X[i]的数字对输入字符串（之后简称s）的第i位加密。加密方法为：循环30次（j=0~29），若j为奇数，将s和X异或；否则s加上X的值。然后令X=X\*97+7。因此解密也十分简单，任意得到某个整数下的加密结果，可以构造出数组X，暴力枚举每一个字符在每一位的加密结果并验证。得到\*ctf{WOW\_i5\_this\_c1pher\_rea1ly\_complicated??!}

**sh**:运行程序后发现这个程序几乎是一个sh，但是反编译检查代码后发现其输出当前目录地址的函数0x401193在当绝对目录名第2个字母是p时会调用进程0x400FA2。这个程序检查了是否第3 4 5个字母分别为roc，也就是说这段代码在当前目录处于/proc/时会调用。然后进程获取proc的下一层目录并转成十进制数字d。当满足d^752593%3138295297==2101783863时，输出\*ctf{str(x)}。所以只需要解出方程即可。可以发现模的这个数等于50891\*61667，显然可以得到d^752593%50891==2101783863%50891, d^752593%61667==2101783863%61667。枚举d解出两个方程在d<50891和d<61667的解，就可以知道d%50891=9122和d%61667=27567，解得d=314159265，得到\*ctf{314159265}

**leakage**:运行代码，提示你程序运行空间经过了随机化。之后会有输入名字，并询问main函数所在的地址。输入名字发现，除了输入的信息在其后会跟着一些乱码。反编译源码后发现使用了read进行读取，并不会在后面添加\0，同时栈空间有着一些残留信息，为之前在栈上出现的内容。通过这个信息，我们可以控制名字输入长度来获取从buffer地址开始至多144个字节的信息。虽然地址经过了随机化，但是在执行代码时，非库函数间代码的相对位置是不变的。同时，同一个程序在栈上的行为也是确定的，因此在本地运行leakage，通过pmap检查程序地址空间，调整输入名字的长度来找到残留在栈空间的某一个非库函数的地址，然后计算偏移量算出main函数的地址。得到\*ctf{ithinkyoudbetterinitalizeyourstack}

**ha?**:I’m not happy with haskell. 经过猜测，发现需要输入一个大整数D，满足d%arr[i]=t[i]。使用中国剩余定理求解，得到解为172422282955526557272420239733996943438305195882249154057182238824896714153729386270058950137909604957537166221848663721197156751566234144960275930080154385567483899397625311698926178394763250035442862353002682792293201635194627062199951208188367133165847351849183776402598921549002077207777900223672050874，调用系统指令得到kiBlTGpe/vj3b/4dRSJW/T2XQzHABA4V18ge2VztkO8=，与密文异或得到\*ctf{84e4b84cc4e38df2d96267ffe35241af}

**c++11 and lambda**:首先将下划线们替换成func1-func6, l1-l6, e1-e7。求值时传入的匿名函数return -(~i)即为i+1。经过理解发现l1-l6的函数是返回一个将传入函数连续执行i次的函数。例如：l4(func f) -> return { f’(x) return f(f(f(f(x)))) }。然后没有研究e的内容，开始暴力枚举，发现e4-e7的值和a，b线性相关。得到\*ctf{cxx11\_i5\_fUn(19260817)}

**blog: flag2**:由于shIr0kawa说这是在学习git和github创建的博客，所以这个博客很可能是用git进行版本控制的。首先搜索hexo包含的文件，尝试访问/blog/\_config.yml，发现可以获取文件，说明外部访问权限不止public文件夹，但是访问目录则会403forbidden。尝试后发现/.git出现403forbidden，并且可以访问其下的文件。访问/.git/config，在其中发现 url = https://shIr0kawa:e6d33e42109f91917a50d4c1e44926accbbeb7b6@github.com/starctf/starctf\_blog.git。将这个仓库clone下来，发现是一个私人仓库，包含了这个博客的所有信息。查找History，发现有一个blog & flag deployed的版本，在flag.php中得到\*ctf{.git\_Sh0U1d\_N0T\_8E\_3Xpo5eD}

**daoke**:查看/root/.ash\_history，发现黑客安装了python，更换了apk的源，设置DNS服务器，进入了一个被隐藏的目录，将ctf.fudan.edu.cn:80的内容导入到secret文件中。查找发现secret文件位于/var/tmp/下，内容为一个请求网址无法获取的错误网页。显示访问的地址是http://202.120.224.114/backdoor。将ip换成域名后，发现跳转至百度，检查标头，发现状态码302跳往百度。在标头中得到\*ctf{d0ck3r\_i5\_v3ry\_1n7er3Stin9}

**sql**:装了sqlite执行explain和substr了解输出内容，然后发现每次调用substr都为一个字符，且前一次调用的第二个参数代表该字符在字符串的位置。将字符连接起来后得到\*ctf{S1MpLe\_sqlIte\_bYtec0de}

**blog: flag3**:在stage2中成功clone了一个private的仓库，因此使用此token是有权利clone该用户能够访问的所有仓库的，只需要得到仓库的名称就可以clone。谷歌搜索以后发现github提供了查询用户所有repository的API，通过https://api.github.com/user/repos查询到仓库名称/starctf/starctf\_blog\_stage3\_d6c75c60cc80e1615e3f.git，将其clone下来，在README.md中得到\*ctf{U\_kN0W\_g1Thu8\_V3ry\_W3!!}